UvA Trilearn 2001 Research Abstract

R. de Boer, J.R. Kok, F.C.A. Groen
{remdboer,jellekok,groen}@wins.uva.nl

Faculty WINS, Department of Computer Science
Intelligent Autonomous Systems Group
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

Abstract

This short paper describes the research focus of the *UvA Trilearn 2001* simulation soccer team. This team is built by two masters students from the Intelligent Autonomous Systems Group at the University of Amsterdam for their graduation project. *UvA Trilearn 2001* is a new team that will participate for the first time in RoboCup 2001. For several reasons described below we have chosen not to copy any code from past RoboCup teams. Our initial focus has therefore been on lower-level aspects and software engineering issues. The results so far have been very encouraging. Long-term scientific goals will focus more on higher-level aspects such as multi-agent modeling using a coach agent, adversarial planning and the use of heterogeneous agents.

1 Introduction

In the past the University of Amsterdam has been successful in the RoboCup simulation competition with the team *Windmill Wanderers*, which became 3rd in 1998. Sadly the creator of this team, Emiel Corten, deceased in 1999 and as a result the project came to a halt. By the end of the year 2000 however we were asked to build a new soccer simulation team called *UvA Trilearn 2001* for our graduation project. For two reasons we decided not to continue with the semi-finished *Windmill Wanderers* team or copy any code from other past RoboCup teams. Firstly, we found that the released source codes did in no way conform to regular software standards, i.e. they were poorly structured and poorly documented. We therefore felt that it would be more time-consuming to reuse the code and restructure it into the architecture that we desired than it would be to write it all ourselves. Secondly, we also felt that the lower-levels used by some of the top teams from recent years could be improved in several ways. We thus decided to start from scratch. Our initial focus has been on the lower-levels of our architecture which are essential for the team as a whole. We have successfully tried to improve upon some of the methods used by top teams from the past leading to an advanced low-level implementation. Using a simple high-level decision module on top of these lower levels we have already succeeded in beating the RoboCup 2000 champion FC Portugal. Since it is our intention to carry this project over to others after our graduation, much effort has also been spent on software engineering issues to facilitate future use. Currently the lower levels of our architecture are almost finished. From now on we will therefore focus more on higher-level aspects including adversarial planning, multi-agent modeling using a coach agent and the use of heterogeneous agents.

2 Initial Research Focus

Since we have started from scratch the initial focus has been on the lower levels of our architecture and on software engineering issues. Although it is considered to be scientifically more prestigious to work on high-level behaviour, it is our opinion that the lower levels are the most crucial for the success of the team. Furthermore low-level imperfections cannot be compensated for by high-level behaviour added later on. We thus realized the significance of the low-level implementation and felt that it was worth spending a lot of time on. We have tried to improve upon some of the low-level methods used by top teams from the past. This has among other things led to an advanced synchronisation scheme and more accurate methods for position and velocity estimation. Below we will shortly describe only our synchronisation method and software engineering approach for space reasons. For a more detailed account of our current implementation (including our multi-threaded three-layer architecture, predictive world model and fast-play strategy) we refer to our team description paper.

2.1 Synchronisation

We use an advanced synchronisation method which for each cycle determines the optimal moment to send an action to the server. This moment depends on the arrival times of the different server messages and thus changes from each cycle to the next. Furthermore our method guarantees that the action sent is always based on the latest information from the server. After sending an action it is checked whether the server has actually performed it. When this is not the case (i.e. the action did not arrive in time) no action is sent in the next cycle to avoid a clash (two actions in one cycle). Experiments have shown that our synchronisation method clearly outperforms several others used in the past. During a full-length match consisting of 6000 simulator cycles the number of holes (no action during a cycle) for each agent is less than 0.3% of the total number of cycles and the number of clashes is less than 0.01%

2.2 Software reusability

In order to make it easy for others to use our code after our graduation much attention during implementation has been focused on software engineering issues. We have consequently followed an object oriented design methodology leading to a logical class hierarchy and highly modular code. We have chosen to use the programming language C++ for performance reasons and our current implementation consists of 17000 lines. We have developed a multi-level log system for debugging purposes. This system enables the programmer to specify the abstraction level from which he desires information thereby speeding up the debugging process. Furthermore a strict version management protocol is used to support concurrent development. Much time has also been spent on extensively documenting our code to facilitate future reuse. For this we have used the documentation system Doxygen.

3 Future plans

Since our lower levels are currently almost finished we will from now on focus our attention more on higher-level issues. Our future research will mainly focus on multi-agent modeling using a coach, adversarial planning and the use of heterogeneous agents. Some of our future plans are described below. By the time RoboCup starts we expect to have these implemented.

- Heterogeneous agents and dynamic resource allocation. What is the best way to divide tasks between agents given the fact that they have different skills and that they grow tired? The latest version of the soccer server offers seven different player types each with their own strengths and weaknesses. One can assess the quality of each player type using a formula which returns a utility value based on the player parameters. It is then possible to determine which player types are best suited for which roles inside a formation using a machine learning technique such as a genetic algorithm.
- Multi-agent modeling using a coach agent. In our current implementation individual agent decisions are often affected by actions from opponents, but strategy decisions for the team as a whole are mostly fixed. Ideally however the team strategy should be adjusted in response to adversary behaviour. We want to use the coach agent to analyse the strengths and weaknesses of the enemy team and to give advice on the best possible strategy. To this end the coach must be able to classify enemy behaviour based on certain features. He can then decide which strategy is most appropriate for the given behaviour class. The coach can for instance determine on which side of the field an enemy attack is most dangerous and recommend to change the formation or switch player types for a certain position inside the current formation. In turn the players must know how to interpret the coach's advice and how to act upon it.

4 Conclusion

In this paper we have discussed the research focus and future plans of the *UvA Trilearn 2001* soccer simulation team. Our initial focus has been on the lower levels in the architecture and on software engineering issues. We want to emphasize the fact that software reusability is very important and strongly encourage everyone to spend more time on properly structuring and documenting their code. This will facilitate future use by other researchers inside as well as outside their own community. Our lower levels are currently almost finished. We have tested them by playing a simplified version of the RoboCup 2000 champion *FC Portugal*. This team uses a simple high-level decision module that we also used on top of our own lower levels. Over the course of 10 games we managed to beat this team by a total score of 20-17. We can thus conclude that our lower layers are at least as good as those of *FC Portugal*. Since the qualification deadline for Seattle was much sooner than we expected we were then forced to quickly start working on our higher levels. We have extended our simple decision module to encompass an effective fast-play strategy (including strategic positioning, opponent marking and depth-passing) and already succeeded in beating *FC Portugal* on a couple of occasions. Our future research will focus on the issues shortly discussed in the previous section. We expect to have these implemented by the time the RoboCup starts in august. *UvA Trilearn 2001* will also participate in the German Open in the beginning of june. This will serve as a testcase for Seattle. We refer to our team description paper for a more detailed description of our current implementation. Since many enhancements are still possible at this moment we are currently very happy with our results and feel that we have achieved a lot in a short time.